METHOD FOR CONTROLLING INTERRUPTS AND AUXILIARY CONTROL CIRCUIT

Field of the Invention

[0001] This present invention relates generally to microprocessor systems, and, more particularly, to a method and an auxiliary circuit for controlling interrupts.

Background of the Invention

[0002] While a microprocessor is working, it may be necessary to interrupt the program it is running to execute certain instructions. This is done by using flag signals called interrupts or interrupt requests. An interrupt controller receives these flag signals from peripherals and, depending on the received interrupt, sends to the processor an interrupt command and an interrupt pointer vector. The pointer vector specifies a memory location at which a relative interrupt service routine (ISR) to be run is stored.

[0003] The microprocessor then suspends the operation in progress, saves the state of the program being run so that it may resume the program later, and carries out the instructions of the ISR routine relative to the received interrupt. When the ISR routine has been performed, the processor restores the state of the program and, if no other interrupt is pending, resumes the program from the point at which it was interrupted.

[0004] Interrupt controllers typically have priority registers for establishing which interrupt, among a plurality of received interrupts that are pending, are to be served first. A basic prior art interrupt control circuit or controller with priority ratings is illustrated in FIG. 1. The interrupt flags, INT0, ..., INTk, which are generated by peripherals, are stored in a pending interrupt register INT PENDING REG.

[0005] The block IRQ MASK AND PRIORITY LOGIC includes a so-called mask of interrupts and a priority logic circuit. The priority logic circuit generates an interrupt request signal IRQ REQ and stores its relative priority HIGHEST PRIORITY INT in the register CURR IRQ PRIORITY REG.

[0006] A circuit HIGHEST PRIORITY INT (illustrated with a dashed box) is for processing interrupt request signals based upon the priority thereof. Its core is a state machine IRQ SM which receives as an input an interrupt request signal, and it generates an interrupt command nIRQ for the microprocessor. The interrupt request signal IRQ REQ selects an interrupt vector IRQ VECTOR corresponding to the interrupt to be served, which is read from an interrupt table IRQ VECTOR REG that stores interrupt vectors identifying corresponding ISR routines.

[0007] The register CURR IRQ PRIORITY REG and the register INT PRIORITY STACK allow so-called nested interrupts. The register CURR IRQ PRIORITY REG stores the priority of the currently served interrupt. If an interrupt with a higher priority rating is generated, servicing of the first interrupt is suspended and the relative priority is stored in the register INT PRIORITY STACK, while the new interrupt of higher priority is served and its priority is stored in the register CURR IRQ PRIORITY REG.

[0008] Once the second interrupt of higher priority has been served, the previously suspended interrupt is resumed, if it has a higher priority than any other pending interrupts. Once it has been served, it is deleted from the stack INT PRIORITY STACK by a command STACK PUSH/POP of the state machine IRQ SM.

[0009] The interrupt control circuit has a limited number of input pins dedicated for receiving interrupts from peripherals. Therefore, as illustrated in FIG. 2, only a few peripherals can avail themselves of the dedicated input pins of the interrupt control circuit, while other peripherals share a common input pin for all their interrupts.

[0010] As shown in FIG. 3, the peripheral A is connected to the interrupt control circuit INTERRUPT CONTROLLER such that each possible interrupt signal corresponds to a dedicated pin of the control circuit. By contrast, the peripheral B may use only one pin of the interrupt control circuit INTERRUPT CONTROLLER for all of its interrupts.

[0011] For peripheral B, the interrupt control circuit receives an interrupt signal IRQm that is obtained by ORing the interrupts stored in the interrupt pending register of the peripheral. When the signal IRQm is active, the interrupt control circuit sends to the microprocessor an interrupt command nIRQ and an interrupt vector IRQVECTOR identifying a specific interrupt service routine ISR. This results in the reading of the interrupt pending register of the peripheral B for identifying the requested interrupt, and the eventual serving thereof. Of course, this burdens the processor in executing the ISR routine because it has to read the interrupt pending register of the peripheral B before serving the interrupt.

Summary of the Invention

[0012] In accordance with the present invention, a method is provided for controlling interrupts which allows the microprocessor to quickly know which interrupt is to be served, even if the control circuit receives an interrupt signal resulting from ORing different interrupt flags.

[0013] The method of the invention includes generating a bit string which identifies the position of an active bit in the interrupt pending register of the peripheral requesting the interrupt, and sending the bit-string to the processor. In this way, the processor is able to substantially immediately recognize which interrupt of the peripheral is to be served or processed without having to read the interrupt pending register of the peripheral.

[0014] More particularly, a method for controlling interrupts generated by a peripheral in accordance with the present invention may include storing, in an interrupt pending register, active bits corresponding to interrupt flags generated by the peripheral. Further, an interrupt signal resulting from ORing the interrupt flags may be sent to an interrupt control circuit coupled to the peripheral. When the interrupt control circuit receives the interrupt signal, the interrupt signal may be identified and served.

[0015] In accordance with the invention, identification of the interrupt to be served is significantly less burdensome on the processor that executes the corresponding routine. This is done by generating in the peripheral a bit string identifying, by a corresponding active bit of the string, the interrupt to be served, and sending this bit string to a processor that will so recognize the interrupt based thereon and eventually serve it by executing the corresponding routine.

[0016] The above-described method may be implemented by an auxiliary control circuit that includes an encoding circuit for encoding in a bit string the position of an active bit stored in the interrupt pending register of the peripheral that generated an interrupt flag. The auxiliary control circuit may then send the bit string to the processor. By way of example, the auxiliary control circuit may be implemented in a dedicated device to which a plurality of peripherals are coupled, or it may be implemented in each one of the peripherals.

Brief Description of the Drawings

[0017] The various aspects and advantages of the present invention will become more apparent through a detailed description thereof, with reference to the accompanying drawings, in which:

[0018] FIG. 1 is schematic circuit diagram of a basic prior art interrupt control circuit;

[0019] FIG. 2 is a schematic circuit diagram of a prior art microprocessor system in which certain peripherals send interrupts to the microprocessor via dedicated pins of the control circuit, while the interrupts of other peripherals share a single pin;

[0020] FIG. 3 is a schematic circuit diagram of two interrupt pending registers connected in different ways to an interrupt control circuit in accordance with the prior art;

[0021] FIG. 4 is a schematic circuit diagram illustrating a first embodiment of an auxiliary interrupt control circuit in accordance with the present invention connected to a plurality of peripherals;

[0022] FIG. 5 is a schematic circuit diagram illustrating a second embodiment of an auxiliary interrupt control circuit in accordance with the present invention including interrupt mask circuits;

[0023] FIG. 6 is a schematic circuit diagram illustrating another embodiment of an auxiliary interrupt control circuit in accordance with the present invention including a RAM for defining the priority levels of interrupts;

[0024] FIG. 7 is a schematic circuit diagram illustrating a microprocessor system including an auxiliary interrupt control circuit in accordance with the present invention; and

[0025] FIG. 8 is a schematic circuit diagram of a peripheral including an auxiliary interrupt control circuit in accordance with the present invention.

Detailed Description of the Preferred Embodiments

[0026] Referring now to FIG. 4, a first embodiment of the auxiliary interrupt control circuit AUXILIARY IC in accordance with the present invention is first described. This circuit may be implemented as a device distinct from the peripherals and from the interrupt control circuit INTERRUPT CONTROLLER, and it is connected to a plurality of peripherals through the bus DATA BUS as shown.

[0027] The auxiliary circuit AUXILIARY IC includes an encoding circuit ENCODER that generates a bit string. The bit string identifies the position of an active bit stored in the interrupt pending register of the peripheral that required the interrupt, and it corresponds to the interrupt to be served. This bit string is sent to the microprocessor CPU which knows substantially immediately which interrupt is to be served based thereon. This is so even if the control circuit received an interrupt signal resulting from ORing different interrupt flags. Therefore, the processor need not read the contents of the interrupt pending register of the peripheral that has requested the interrupt. Instead, it may serve it directly.

[0028] The auxiliary circuit AUXILIARY IC has an auxiliary register TEMP INT PENDING REG for use in the encoding operation. When an interrupt IRQ is received by the interrupt control circuit, the microprocessor CPU identifies the peripheral it belongs to and copies the entire contents of the relative interrupt pending register PERIPHERAL INT PENDING REG in the auxiliary register TEMP INT PENDING REG.

[0029] The encoding circuit ENCODER generates a bit string that identifies the position of a first active bit stored in the auxiliary register and sends it to the microprocessor through the register FIRST ONE REG and the bus DATA BUS. In the illustrated example, the connected peripherals have 32-bit registers for storing pending interrupts, thus a 5-bit string is sufficient for encoding the position of the first active bit.

[0030] Optionally, the encoding circuit ENCODER may also generate a second bit string for encoding the last active bit stored in the auxiliary register, and even a third bit string for encoding the number of active bits stored therein. These last two strings are stored in respective dedicated registers LAST ONE REG and NUMBER

OF ONES REG, before being sent to the microprocessor.

[0031] As illustrated in FIG. 5, the auxiliary circuit of the present invention may have priority interrupt masks INT MASK and INT PRIORITY MASK, respectively. Here, the first active bit identified by the string generated by the encoding circuit ENCODER corresponds to the interrupt with highest priority. For example, assuming that the auxiliary register stores the following string:

0001 0010 0011 1100 1000 1000 0000 0000,

and the priority interrupt mask(s) assigns a higher priority to the bits from the ninth to the twentieth than to the other bits, the encoding circuit ENCODER will identify the eleventh bit with the first string, because it is the first active bit having the highest priority.

[0032] In the case where the auxiliary register stores the following string:

0001 0010 0000 0000 0000 1000 0000 0000,

and the priority interrupt mask(s) is the same as in the previous example, the first string of the circuit ENCODER will identify the fourth bit.

[0033] According to another embodiment illustrated in FIG. 6, the auxiliary interrupt control circuit of the present invention illustratively includes a writable memory RAM PRIORITY MASK that stores priority values for configuring, from time to time, the priority interrupt mask(s). It is generally desirable to embody the auxiliary circuit of the present invention in the interrupt control circuit. This allows the same memory that is already present in a common interrupt control circuit INTERRUPT CONTROLLER to be shared for the same functions. This is particularly convenient given that the memory of the interrupt control circuit that stores priority values is periodically updated. Having a memory RAM PRIORITY MASK in the auxiliary control circuit as well would require a duplication of the updating.

[0034] The auxiliary interrupt control circuit AUXILIARY IC in accordance with the present invention may be implemented as a separate device that is coupled to a plurality of peripherals, as illustrated in FIG. 7. It may also be incorporated in a respective peripheral for providing to the interrupt control circuit an interrupt signal corresponding to the logic OR of different interrupt flags. This embodiment is illustrated in FIG. 8. Here, the peripheral illustratively includes the interrupt pending register INT PENDING REG and the peripheral circuits PERIPHERAL KERNEL. The block PERIPHERAL KERNEL generates the interrupts flags INT0, ..., INTn that are stored in the interrupt pending register INT PENDING REG, while an OR gate generates the interrupt signal IRQm that is provided to the control circuit INTERRUPT CONTROLLER by ORing the various interrupt flags.

[0035] The auxiliary control circuit reads the interrupt pending register and generates, at a minimum, a code that identifies a first active bit in the register INT PENDING REG and sends it to the processor through the bus DATA BUS. Optionally, the auxiliary circuit embodied in the peripheral may also include a priority mask(s) as discussed above with reference to FIG. 5.